Polecenie: Napisać prosty generator liczb pseudolosowych według przepisu: Rn+1 = [75 · (Rn + 1) mod 65537] − 1. Dokładnie taki generator był stosowany na popularnych w latach 80-tych komputerach ZX Spectrum, a generowane liczby zawierają się w przedziale 0, . . . , Rmax = 2 16 − 1. Można je łatwo konwertować na liczby zmiennoprzecinkowe z przedziału [0, 1) z pomocą instrukcji rzutowania: x = (double)R / (R_MAX + 1.0). Następnie, proszę wykorzystać opisany algorytm do generacji dużej liczby (n ∼ 102) par punktów na płaszczyźnie (x, y) należących do kwadratu {0 6 x < 1 ∧ 0 6 y < 1}. Wiedząc, że prawdopodobieństwo trafienia w koło {x 2 + y 2 < 1} wynosi π/4, wyznaczyć przybliżoną wartość liczby π na podstawie ułamka k/n, gdzie k oznacza liczbę trafień we wnętrze koła.

Tym razem nie będę skupiał się na rozpisywaniu działania programu a na badaniu jego wyników. Ale na wstępie krótki opis:

Na początku ze wzoru podanego w poleceniu zrobiłem funkcję generującą losową liczbę. Następnie zastosowałem wzór x^2+y^2<1 i aby mieć 2 różne wartości x i y do wartości y przy Rmax dodałem 1 (lub inną wartość) tak aby liczby się różniły.

==========

Badanie wyników programu:

Dla liczby R początkowe=15:

```
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ gcc zad5.c -o zad5
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10
2.800000
7.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100
2.840000
71.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 1000
3.140000
785.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10000
3.134000
7835.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100000
3.141400
78535.000000
```

Widać że im większe n tym wynik jest coraz bliższy liczbie pi.

Dla liczby R początkowe=150:

```
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10
3.200000
8.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100
2.960000
74.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 1000
3.164000
791.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10000
3.136400
7841.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100000
3.141840
78546.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$
```

Łatwo zauważyć że dzieje się podobnie jak w poprzednim przykładzie lecz tutaj liczby są trochę bardziej różne od liczby pi, niewątpliwie oscylują w jej wartości lecz nie trafiają tak dokładnie jak było to poprzednio.

Dla liczby R początkowe=1550:

```
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10
3.200000
8.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100
2.920000
73.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 1000
3.028000
757.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10000
3.142800
7857.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100000
3.139760
78494.000000
```

Tutaj natomiast widać że zasada poniekąd pokrywa się z poprzednimi przykładami czyli im większe n tym bliżej 3.14.. lecz w porównaniu do R=150 widać że rozbieżność jest jeszcze większa.

Wniosek chwilowy: wydaje się że wpływ na dokładność wyniku przede wszystkim ma ilość zastosowanych prób czyli zmienna n. Łatwo zauważyć że dla n=10 wynik jest najmniej prawidłowy a dla n=1000-100000 wynik jest już w okolicach liczby 3.14. Natomiast co ważniejsze i mniej oczywiste im większe R początkowe tym wynik jest coraz bardziej rozbieżny od wartości liczby Pi.
Wydaje mi się że najbardziej stosowny jest tutaj przykład 1 czyli R=15 dlatego że 2 na 5 prób dały wynik 3.14…. a pozostałe próby czyli R=150 oraz R=1550 dały chciany wynik tylko 1 na 5 prób.

==========

Teraz zamiast zmieniać R początkowe zmienię także liczby jakie dodaje do x i y (tak aby liczby zrobiły się różne):

Dla R początkowego =15 ale do x+15 || y+325

```
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ gcc zad5.c -o zad5
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10
2.400000
6.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100
2.960000
74.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 1000
3.040000
760.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10000
3.140400
7851.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100000
3.142760
78569.000000
```

Widać że zmiana x I y miała faktycznie znaczący wpływ na wynik, dopiero powyżej n=10000 wynik zaczął być równy 3.14….

Dla R początkowego =150 ale do x+15 || y+325

```
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10
2.400000
6.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100
2.880000
72.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 1000
3.084000
771.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 10000
3.141600
7854.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5
Podaj liczbe n: 100000
3.142840
78571.000000
```

Jak łatwo zauważyć pomiędzy tym a poprzednim przykładem nie ma praktycznie różnic, ale z poprzednich przykładów widać że większe zróżnicowanie wartości x i y dobrze wpłynęło na wynik końcowy, z 2 na 2 przykłady oba miały 2/5 wartości równych 3.14…..

WNIOSEK OGÓLNY:

Z powyższych przykładów i badań zauważyłem że wartości otrzymane nigdy nie będą dokładną wartością liczby pi. Nawet przy zastosowaniu n=10^9999999 wynik dalej będzie się różnił z kolejnymi liczbami po przecinku liczby pi. Natomiast zastosowanie różnych R początkowych miała niewątpliwie duży wpływ na wynik. Wydaje się że im mniejsze R (np. R=15) wynik jest bardziej poprawny niż przy zastosowaniu dużej wartości R.

==========

ULEPSZONA WERSJA PROGRAMU Z FUNKCJĄ rand():

```
srand(time(NULL));
for (i = 1; i <= n; i++)
{
x = rand() / (double)RAND_MAX;
y = rand() / (double)RAND_MAX;
if (x * x + y * y <= 1)
k++;
```

Dzięki zastosowaniu tej funkcji program działa nieco inaczej bo w oparciu o czas. Wyniki też są zgoła inne:

```
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ gcc zad5ulepszone.c -o zad5u
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5u
Liczba losowanych punktow n = 10
3.200000
8.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5u
Liczba losowanych punktow n = 100
3.080000
77.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5u
Liczba losowanych punktow n = 1000
3.164000
791.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5u
Liczba losowanych punktow n = 10000
3.142000
7855.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5u
Liczba losowanych punktow n = 100000
3.136880
78422.000000
bartosz@Baxing-VB:~/Desktop/C/Zestaw2$ ./zad5u
Liczba losowanych punktow n = 1000000
3.143564
785891.000000
```

Widać że liczby już dla n=10 są praktycznie bliskie liczbie pi, natomiast ze znacznym wzrostem liczby n np. 100000 liczba jest już praktycznie liczbą pi (lecz trzeba pamiętać że w dalszej części okresowości liczby pi różnice będą znaczące). Z punktu użytkownika warto też zauważyć że program z funkcją rand() działa dużo szybciej niż generowanie liczb pseudolosowych algorytmami.